Computer vision Seminar

수원대학교
Data Network Analysis

데이터과학부
정인호

2023.03.15

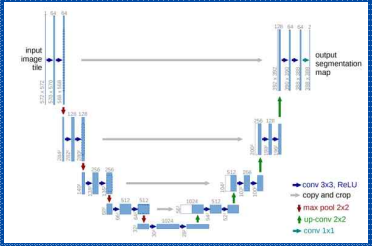
Week8) CNN Architecture 논문 리뷰

1. 지난 주 세미나 정리
2. Dense Net (2016)
3. CNN Attention – SE Net(2017)
4. 깃헙 정리

수원대학교
Data Network Analysis

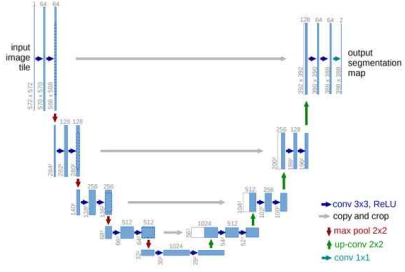
Week8) CNN Architecture 논문 리뷰

1. 지난주 세미나 정리



수원대학교
Data Network Analysis

1. 지난 주 세미나 정리


input image tile
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

Week8) CNN Architecture 논문 리뷰

2. DenseNet (2016)

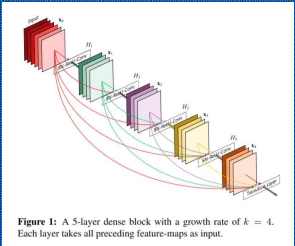


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

2. DenseNet (2016)

논문 링크
논문 초록을 읽자

Abstract

Recent work has shown that convolutional networks can be substantially deeper, more accurate, and efficient to train if they contain shorter connections between layers close to the input and those close to the output. In this paper, we embrace this observation and introduce the Dense Convolutional Network (DenseNet), which connects each layer to every other layer in a feed-forward fashion. Whereas traditional convolutional networks with L layers have L connections—one between each layer and its subsequent layer—our network has $\frac{L(L+1)}{2}$ direct connections. For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. DenseNets have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters. We evaluate our proposed architecture on four highly competitive object recognition benchmark tasks (CIFAR-10, CIFAR-100, SVHN, and ImageNet). DenseNets obtain significant improvements over the state-of-the-art on most of them, whilst requiring less computation to achieve high performance. Code and pre-trained models are available at https://github.com/liuzhuang13/DenseNet.

2. DenseNet (2016)

선행 연구 ResNet

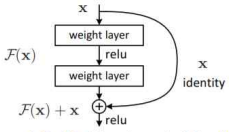


Figure 2. Residual learning: a building block.

DenseNet

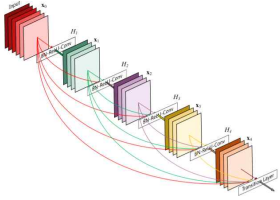


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

1. Introduction ...

In this paper, we propose an architecture that distills this insight into a simple connectivity pattern: to ensure maximum information flow between layers in the network, we connect *all layers* (with matching feature-map sizes) directly with each other. To preserve the feed-forward nature, each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers. Figure 1 illustrates this layout schematically. Crucially, in contrast to ResNets, we never combine features through summation before they are passed into a layer; instead, we combine features by concatenating them. Hence,

2. DenseNet (2016)

문제정의

Recent variations of ResNets [13] show that many layers contribute very little and can in fact be randomly dropped during training. This makes the state of ResNets similar to (unrolled) recurrent neural networks [21], but the number of parameters of ResNets is substantially larger because each layer has its own weights. Our proposed DenseNet architecture explicitly differentiates between information that is added to the network and information that is preserved. DenseNet layers are very narrow (*e.g.*, 12 filters per layer), adding only a small set of feature-maps to the "collective knowledge" of the network and keep the remaining feature-maps unchanged—and the final classifier makes a decision based on all feature-maps in the network.

each layer has its own weights. Our proposed DenseNet architecture explicitly differentiates between information that is added to the network and information that is preserved. DenseNet layers are very narrow (*e.g.*, 12 filters per layer), adding only a small set of feature-maps to the "collective knowledge" of the network and keep the remaining feature-maps unchanged—and the final classifier makes a decision based on all feature-maps in the network.

선행 연구 ResNet

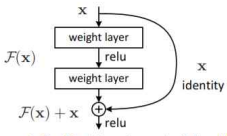


Figure 2. Residual learning: a building block.

DenseNet

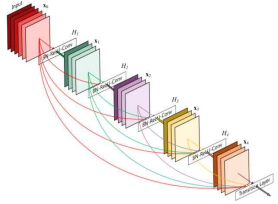


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

2. DenseNet (2016)

의의

Instead of drawing representational power from extremely deep or wide architectures, DenseNets exploit the potential of the network through *feature reuse*, yielding condensed models that are easy to train and highly parameter-efficient. Concatenating feature-maps learned by *different layers* increases variation in the input of subsequent layers and improves efficiency. This constitutes a major difference between DenseNets and ResNets. Compared to Inception networks [36, 37], which also concatenate features from different layers, DenseNets are simpler and more efficient.

선행 연구 ResNet

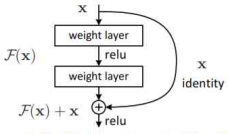


Figure 2. Residual learning: a building block.

DenseNet

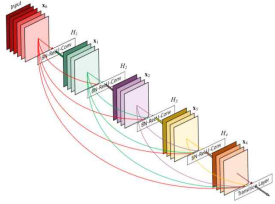


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

2. DenseNet (2016)

DenseNet 아키텍쳐 구조

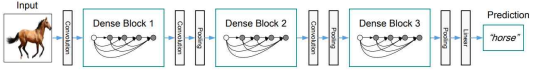


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

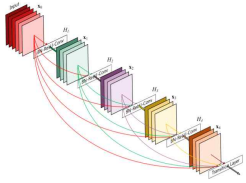


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

2. DenseNet (2016)

DenseNet 아키텍쳐 구조

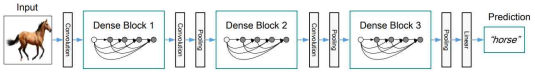


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Table 1: DenseNet architectures for ImageNet. The growth rate for all the networks is $k = 32$. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.

Pooling layers. The concatenation operation used in Eq. (2) is not viable when the size of feature-maps changes. However, an essential part of convolutional networks is down-sampling layers that change the size of feature-maps. To facilitate down-sampling in our architecture we divide the network into multiple densely connected *dense blocks*; see Figure 2. We refer to layers between blocks as *transition layers*, which do convolution and pooling. The transition layers used in our experiments consist of a batch normalization layer and an 1×1 convolutional layer followed by a 2×2 average pooling layer.

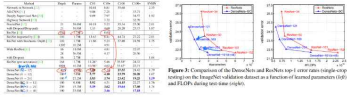
Growth rate. If each function H_ℓ produces k feature-maps, it follows that the ℓ^{th} layer has $k_0 + k \times (\ell - 1)$ input feature-maps, where k_0 is the number of channels in the input layer. An important difference between DenseNet and existing network architectures is that DenseNet can have very narrow layers, *e.g.*, $k = 12$. We refer to the hyper-parameter k as the *growth rate* of the network. We show in Section 4 that a relatively small growth rate is sufficient to obtain state-of-the-art results on the datasets that we tested on. One explanation for this is that each layer has access to all the preceding feature-maps in its block and, therefore, to the network's "collective knowledge". One can view the feature-maps as the global state of the network. Each layer adds k feature-maps of its own to this state. The growth rate regulates how much new information each layer contributes to the global state. The global state, once written, can be accessed from everywhere within the network and, unlike in traditional network architectures, there is no need to replicate it from layer to layer.

2. DenseNet (2016)

발표자 블로그

이미지 넷은 위 데이터셋과는 달리 스케일이 굉장히 크기 때문에 table1의 아키텍쳐로 실험을 진행합니다.

Densenet-121만 떄서 보면, growth rate = 32로, 각 dense block 마다 conv block이 2개씩 있기 때문에 파라미터 layer 수를 세보면, 2*(6+12+24+16) = 116 layer입니다. 가장 상단의 7x7 Conv layer 1개, transition layer의 1x1 conv layer 3개, 최하단 1000D fc layer 1개를 더해주어 121 - layer architecture임을 알 수 있습니다.



ResNet에 비해 상당히 좋은 결과를 얻은 걸 알 수 있습니다.

Implicit Deep Supervision. One explanation for the improved accuracy of dense convolutional networks may be that individual layers receive additional supervision from the loss function through the shorter connections. One can interpret DenseNets to perform a kind of "deep supervision". The benefits of deep supervision have previously been shown in deeply-supervised nets (DSN; [20]), which have classifiers attached to every hidden layer, enforcing the intermediate layers to learn discriminative features.

DenseNets perform a similar deep supervision in an implicit fashion: a single classifier on top of the network provides direct supervision to all layers through at most two or three transition layers. However, the loss function and gradient of DenseNets are substantially less complicated, as the same loss function is shared between all layers.

DenseNet AveragePooling의 이유(개인적인 생각)

이 부분은 지극히 개인적인 의견입니다.

보통은 maxpooling을 넣지만 DenseNet 특이하게도 avg pooling을 이용하여 downsmapling을 진행합니다. 자세한 이유가 기재되어 있지는 않습니다.

DenseNet은 앞에서 나온 output을 계속해서 입력값으로 가져오는 feature Reuse방식입니다. feature map에서 어떤 개체가 있거나 하는 아주 강한 픽셀값(큰 값)은 conv를 계속 거쳐도 살아있을 겁니다.

따라서 **같은 사이즈 내에서 계속해서 reuse한 feature map이 concat 되어 있는 output을 max pooling하면 같은 특징들이 뽑힐 확률이 높을 것 같다는 개인적인 생각입니다.**

따라서 maxpooling으로 강한 값만 가져오면 비슷한 특징들이 뽑히니, 각 feature map들의 평균값을 가져오는 방식이 더 나은 결과를 얻어내지 않았을까? 싶습니다.

저자들도 아마 이 부분을 실험을 통해서 얻지 않았을까? 싶은데..

논문 가서 Conclusion부분도 한 번 짚어주기

2. DenseNet (2016)

발표자 블로그

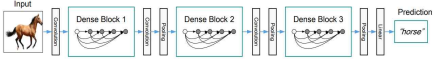


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

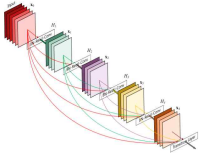


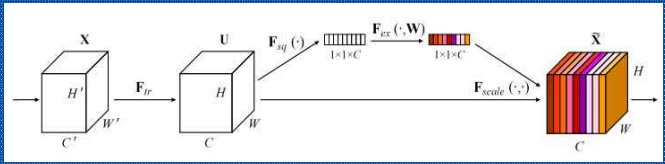
Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

물론 이 모델도 완벽한 것은 아닙니다.

Dense Net 정리

Week8) CNN Architecture 논문 리뷰

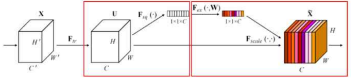
3. Squeeze & Excitation Net(2017)



3. SE Net (2017)

[발표자 블로그]

SE block 구조



SE block 구조도

SE block은 말그대로 Squeeze 과정과 Excitation과정 2단계를 거쳐 channel간의 relationship을 고려하게 됩니다. 먼저 앞에 있는 Squeeze operation을 살펴봅시다.

CNN에서의 Attention이란?

3. SE Net (2017)

Squeeze Operation

Squeeze 작업은 앞 **F tr**(Conv layer)함수에서 받아온 Feature map **U**를 압축하는 것을 목표합니다.

여기서 받아온 Feature map **U**는 각각 학습된 필터들로부터 생성된 특징들인데, 저자들은 이렇게 생성된 각 **채널의 feature map들은 그 region밖에서는 맥락적인 정보로는 이용할 수 없는 점이 문제라고 말합니다.**

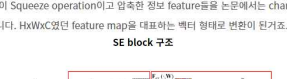
> 여기서 저의 해석은 각 feature map들을 생성한 **F tr** 함수의 filter들은 각자 목표하는 바가 다릅니다.
> 가령 **F tr** 함수가 3x3 conv layer로 (HxWx6)의 feature map **U**를 생성했다고 생각하면,
> 6개의 feature map은 서로 바라보는 관점이 다른 겁니다. 어떤 특징맵은 둥그런 개체에 집중하는 필터를 거친 특징 맵이고, 어떤 개체는 파란색 개체에 좀 더 집중된 특징맵일 수 있습니다.

따라서 이렇게 각각 다른 분야의 전문가들로부터 가져온 정보들이기 때문에 전체 맥락적으로 이용할 수 없으며, 정보 활용이 제한적이라는 것이죠.

그렇기 때문에 feature map U를 Global Average Pooling을 통해 1x1xC로 압축합니다.

이 과정이 Squeeze operation이고 압축한 정보 feature들을 논문에서는 channel descriptor라고 표현합니다. HxWxC였던 feature map을 대표하는 벡터 형태로 변환이 된거죠.

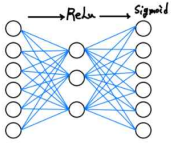
SE block 구조



SE block 구조도

Excitation Operation

이제 압축한 정보들을 전체 맥락적으로 이용하고 싶습니다.



조잡하지만... 제가 예시로 그린 Excitation block입니다.

Squeeze과정으로 1x1x6 벡터가 나왔다고 가정하고 그림을 보시면,

Fc layer를 통해 C/r로 노드수를 줄여 ReLU함수를 거칩니다. (위 경우 C=6, r=2)

다시 FC layer를 통해 최종적으로 C만큼 값을 출력하여 Sigmoid 함수를 통해 0~1의 값을 지닌 6개의 벡터로 만들어줍니다.

왜 굳이 수축을 시키고 그것을 다시 ReLU함수로 넣은 뒤 확률화 시킬까요?

저자들은 채널간에도 반드시 비선형성을 가해주는 것을 의도했다고합니다. 연급했듯이 저자들은 **Channel relationship**에 초점을 맞추었습니다. 따라서 **채널을 한 번 reduction ratio r을 만큼 수축**시킨 뒤 비선형 함수 ReLU로 channel간의 관계를 살핀 것이죠.

(시그모이드 함수로 돌아가기 전 ReLU함수로 0~1값으로 인코딩하여 뒤넣효과를 극대화시키는 부분도 기대해볼 수 있죠.)

그렇기 때문에 채널끼리는 non-mutually-exclusive하게 고려할 수 있게 되는겁니다.

수축시킨 벡터를 다시 C개로 출력하여 마지막으로 시그모이드 함수를 이용해 0~1 값으로 인코딩하여 feature map들을 모두 고려하여 중요도를 매기게 되는 것이고,

최종적으로 이 학률 벡터들을 원래의 feature map U에 곱해주는 것입니다.

결국은 SE block을 통해 feature map U에서 어떤 채널에 집중해줄지 골라주는 역할을 해주는 것입니다.

3. SE Net (2017)

Attention 효과

원래대로라면 각각 다른 전문가들인 filter들로부터 나온 feature map들을 전부 각각 독립적으로 바라보는게 아니라

C개를 전부 고려하여 어떤 feature map이 여기서 얼마만큼 중요한지를 알게해주는 역할을 해준 겁니다.

가령 정말 자세하게 비유하자면, 이번 학습 batch의 이미지는 포도였습니다. 현재 layer에서 C개의 filter 중에 하나는 동그란 계체를 담당하는 필터가 있었고, 보라색을 담당하는 필터도 있었습니다. 두 필터의 시너지를 고려하여 이번 배치는 해당 layer에서 그 두 필터는 SE block에서 높은 점수(1에 가까운 가중치로)를 부여받게 되는 것이죠.

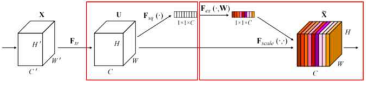
사실 모델이 깊어질수록 좀 더 글로벌한 정보를 뽑고 정보량도 다르기때문에 물체의 모형과 색깔의 비유는 맞지 않을 수 있습니다.

정리하자면, 보통 해당 layer에서 여러 class의 객체들을 구분하는 데에는 n번째랑 k번째의 feature map이 특히 좋더라~ 하고 1에 가까운 값을 부여해주는 것이죠.

SE block이 없었다면, 이렇게 C개의 feature map끼리 고려해 볼 수 있는 과정은 없었을 겁니다.

다시 정리

SE block 구조



SE block 구조도

3. SE Net (2017)

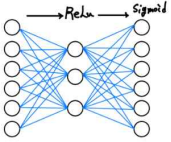
Reduction ratio r

Ratio r	top-1 err.	top-5 err.	Params
2	22.29	6.00	45.7M
4	22.25	6.09	35.7M
8	22.26	5.99	30.7M
16	22.28	6.03	28.1M
32	22.72	6.20	26.9M
original	23.30	6.55	25.6M

사실 하이퍼 파라미터 r을 어떻게 넣어줄지도 고민이 될 수 있습니다.

저자들이 이미지넷 데이터로 SE-ResNet-50에 대해서 r 값을 바꿔가며 얻은 결과들을 공유합니다.

해당 테이블에서 original은 ResNet-50이며, r=8일때의 top-5 에러율이 가장 낮네요.

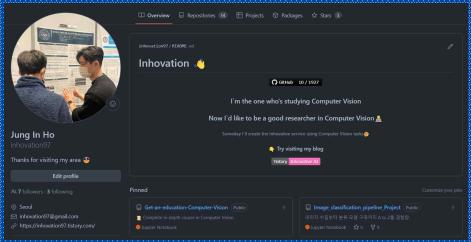


3. SE Net (2017)

Squeeze & Excitation Net 정리

Week8) CNN Architecture 논문 리뷰

4. 깃헙 정리



수원대학교
Data Network Analysis

4. 깃헙 정리

발표자 정인호 깃허브

```
def ComputerVision_Seminar(방학, 과제, 시간, 노력):

    …

    return 깃허브, 코딩실력, 파이토치, 이론지식, 흥미
```

다들 8주간 정말로 수고 많으셨습니다.
잘 따라와 주셔서 너무 감사해요.😁

Good Luck for the own future

수원대학교
Data Network Analysis